Attorney Docket No. 42P18929

APPLICATION FOR UNITED STATES LETTERS PATENT

For

BUFFER VIRTUALIZATION

Inventor(s):

Nicholas Samra
Belliappa Kuttanna
Rajesh Patel

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1030
(408) 720-8300

"Express Mail" mailing label number: EV 409356824 US
Date of Deposit: April 8, 2004
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

Anne Collette
(Typed or printed name of person mailing paper or fee)

Anne Collette
(Signature of person mailing paper or fee)

4/8/2004
(Date signed)

BUFFER VIRTUALIZATION

FIELD

[0001] Embodiments of the invention relate to microprocessor architecture. More particularly, embodiments of the invention relate to a technique for virtualizing register resources within a microprocessor.

BACKGROUND

[0002] High performance microprocessors typically use multi-stage ("deep") pipeline architectures to facilitate running at high frequencies. In order to maintain high instruction parallelism with these deep pipelines, large buffering resources are typically used to minimize stalling of instructions within the pipeline.

[0003] For the example of load operations, a deeply pipelined processor typically has enough load buffers to ensure that, at least most of the time, issuing of new load instructions will not be stalled because all of the available load buffers are currently allocated to un-retired load instructions. This may be true of other operations, such as store operations, as well.

[0004] However, increasing the number of buffering resources may not always be the optimal solution. One reason is that a large buffer structure is more difficult to design than a smaller one. Furthermore, processor performance may be lost if accesses to a large buffer structure are pipelined in order to meet the operating frequency targets.

[0005] Typical high-performance processors are designed with sufficient buffering resources to cover their pipeline depth, at least for the majority of circumstances. Conversely, the pipeline depth can be balanced with the size of buffers that may be successfully implemented at the target frequency. Furthermore, processors with deeper pipelines typically needed more buffers than those with shorter pipelines. Adding more buffers to accommodate deeper pipelines in microprocessors can add cost, increase power consumption, and be difficult to implement.

[0006] In prior art microprocessor architectures, buffer allocation is typically allocated early in the processor pipeline. Therefore, when the physical buffers are allocated, the processor typically stalls the next load instruction (and all subsequent instructions) at the allocate stage of the pipeline until a physical buffer is available. The allocation stage of the pipeline is typically before the scheduling stage of the pipeline in deeply pipelined processors. Consequently, buffer allocation must occur prior to the operations being scheduled, which can degrade processor performance if the pipeline stalls.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Embodiments of the invention are illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0008] Figure 1 is a flow diagram that illustrates the physical buffer check (PBC) algorithm, according to one embodiment, as applied to load operations.

[0009] Figure 2 illustrates the mapping and organization of physical buffers within a virtual buffer file according to one embodiment of the invention.

[0010] Figure 3 illustrates a microprocessor architecture in which one embodiment of the invention may be used.

[0011] Figure 4 illustrates a computer system in which at least one embodiment of the invention may be used.

[0012] Figure 5 is a point-to-point (PtP) computer system in which one embodiment of the invention may be used.

DETAILED DESCRIPTION

[0013] Embodiments of the invention pertain to microprocessor architecture. More particularly, embodiments of the invention pertain to virtualizing physical buffers within a microprocessor.

[0014] The term "buffer" shall be used as a generic term for any computer memory structure, including registers and static random access memory (RAM), and dynamic RAM (DRAM). Furthermore, although numerous references are made to load buffers throughout, the concepts and principals described herein may readily be applied to other types of buffers, including store buffers.

[0015] Buffer virtualization techniques described herein involve increasing the number of allocate-able buffers over the actual number of buffers within or used by a processor in order to facilitate higher processor performance without significantly increasing the cost or complexity of the processor design. For example, a relatively large number of load operations, such as 128 load operations, could be active in the processor at a given time, even though a relatively small number of physical buffers, such as 64 physical load buffers, are actually available.

[0016] In order to increase the effective buffer resources available to a processor architecture, embodiments of the invention involve techniques to map each virtual buffer to a physical buffer when necessary and to ensure that multiple operations, such as load and store operations, that share the same physical buffer entry do not interfere with each other when accessing that physical buffer entry.

[0017] In at least one embodiment of the invention, virtual buffers are mapped to physical buffers by indexing the lower *n* bits of the virtual buffer address into 2^n physical buffer entries. Advantageously, if the number of virtual load buffers is a power of 2 multiple of the physical buffers, for example (e.g. the number of virtual load buffers is 2, 4, 8 etc. times larger than the number of physical load buffers), then each physical buffer can be shared by the same number of virtual load buffers.

[0018] In order to prevent two (or more) load operations that share the same physical buffer entry from interfering with each other when accessing the same buffer, a physical buffer check (PBC) algorithm may be used. Figure 1 is a flow diagram that illustrates the PBC algorithm, according to one embodiment, as applied to load operations.

[0019] After a reset operation that places the processor in a known state, a head buffer pointer (HBP) is set to point to the last physical load buffer at operation 101. When a load buffer is de-allocated, the HBP is incremented by 1, wrapping back to 0 after pointing to the last virtual load buffer entry at operation 105. Whenever a load operation wants to check if the correct physical load buffer is available for it to use, it can check to see if the virtual load buffer index is less than or equal to the HBP (Virtual LB index <= HBP) at operation 110. If the virtual load buffer index is less than or equal to HBP, then the physical load buffer is available at operation 115. Otherwise, the load operation can wait until the HBP is incremented making the above equation true at operation 120.

[0020] Figure 2 illustrates an example of how the PBC algorithm may be used in a processor architecture. In the example illustrated in Figure 2, a stream of load operations are issued within a processor with 64 physical load buffers and 128 virtual load buffers. As illustrated in Figure 2, the physical buffers 201 are a subset of the number of virtual buffers 205. The first 65 load operations are assigned virtual load buffers 0 through 64. The first load operation, which uses virtual load buffer 0, or 0000000 in binary, maps its buffer to the same physical load buffer as the last load operation, which uses virtual load buffer 64, or 1000000 in binary, since the lower 6 binary digits are the same between the two virtual load buffer addresses.

[0021] The HBP 210 would be initialized to 63 in this machine, such that the first load operation will successfully access the load buffer, since the virtual load buffer index is <= HBP, or 0 <=63. However, the last load will fail this check, since the equation, virtual load buffer index <=HBP, will not be true. After the first load operation retires and de-allocates its load buffer, the HBP will increment to 64 and enabling the last load (with virtual load buffer index=64) to access the physical load buffer at index 0.

[0022] The PBC algorithm may be implemented at various stages in the processor pipeline. However, implementing the PTC algorithm at a stage earlier in the pipeline than the stage at which the physical buffer needs to be accessed by an operation, such as a load or store operation, can yield advantageous results.

[0023] Figure 3 illustrates a processor architecture, according to one embodiment, in which the PBC algorithm is implemented in the scheduling stage. Figure 3 illustrates a bus agent in which at least one embodiment of the invention may be used. Particularly, Figure 3 illustrates a microprocessor 300 that contains one or more portions of at least one embodiment of the invention 313, a decoder unit 305, and an allocation unit 310. Further illustrated within the microprocessor of Figure 3 is an execution unit 320 to perform operations, such as store and load operations, within the microprocessor and a retirement unit 325 to retire instructions after they have been executed.

[0024] The PBC algorithm may be implemented partially or completely in logic within any portion of the microprocessor. However, advantageous results can result if the PBC algorithm is implemented in logic within the scheduler unit 315. The exact or relative location of the execution unit and portions of embodiments of the invention are not intended to be limited to those illustrated within Figure 3.

[0025] By implementing the PBC algorithm within the scheduler of the processor in Figure 3, the processor pipeline does not stall at the allocation stage even when all physical buffers are allocated, because operations, such as load and store operations, that do not have a physical load buffer available are simply held in the scheduler until they do. The elimination of allocation stalls can provide processor performance improvement, in at least one embodiment, since other instructions may bypass unallocated operations and be executed.

[0026] Figure 4 illustrates a computer system in which at least one embodiment of the invention may be used. A processor 405 accesses data from

a level one (L1) cache memory 410 and main memory 415. In other embodiments of the invention, the cache memory may be a level two (L2) cache or other memory within a computer system memory hierarchy. Illustrated within the processor of Figure 2 is one embodiment of the invention 406. Other embodiments of the invention, however, may be implemented within other devices within the system, such as a separate bus agent, or distributed throughout the system in hardware, software, or some combination thereof.

[0027] The main memory may be implemented in various memory sources, such as dynamic random-access memory (DRAM), a hard disk drive (HDD) 420, or a memory source located remotely from the computer system via network interface 430 containing various storage devices and technologies. The cache memory may be located either within the processor or in close proximity to the processor, such as on the processor's local bus 407. Furthermore, the cache memory may contain relatively fast memory cells, such as a six-transistor (6T) cell, or other memory cell of approximately equal or faster access speed.

[0028] The computer system of Figure 4 may be a point-to-point (PtP) network of bus agents, such as microprocessors, that communicate via bus signals dedicated to each agent on the PtP network. Within, or at least associated with, each bus agent is at least one embodiment of invention 406, such that store operations can be facilitated in an expeditious manner between the bus agents.

[0029] Figure 5 illustrates a computer system that is arranged in a point-to-point (PtP) configuration. In particular, Figure 5 shows a system where

processors, memory, and input/output devices are interconnected by a number of point-to-point interfaces.

[0030] The Figure 5 system may also include several processors, of which only two, processors 570, 580 are shown for clarity. Processors 570, 580 may each include a local memory controller hub (MCH) 572, 582 to connect with memory 52, 54. Processors 570, 580 may exchange data via a point-to-point interface 550 using point-to-point interface circuits 578, 588. Processors 570, 580 may each exchange data with a chipset 590 via individual point-to-point interfaces 552, 554 using point to point interface circuits 576, 594, 586, 598. Chipset 590 may also exchange data with a high-performance graphics circuit 538 via a high-performance graphics interface 592.

[0031] At least one embodiment of the invention may be located within the memory controller hub 572 or 582 of the processors. Other embodiments of the invention, however, may exist in other circuits, logic units, or devices within the system of Figure 5. Furthermore, other embodiments of the invention may be distributed throughout several circuits, logic units, or devices illustrated in Figure 5.

[0032] Various aspects of embodiments of the invention may be implemented using complimentary metal-oxide-semiconductor (CMOS) circuits and logic devices (hardware), while other aspects may be implemented using instructions stored on a machine-readable medium (software), which if executed by a processor, would cause the processor to perform a method to carry out embodiments of the invention. Furthermore, some embodiments of the invention

may be performed solely in hardware, whereas other embodiments may be performed solely in software.

[0033] While the invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.